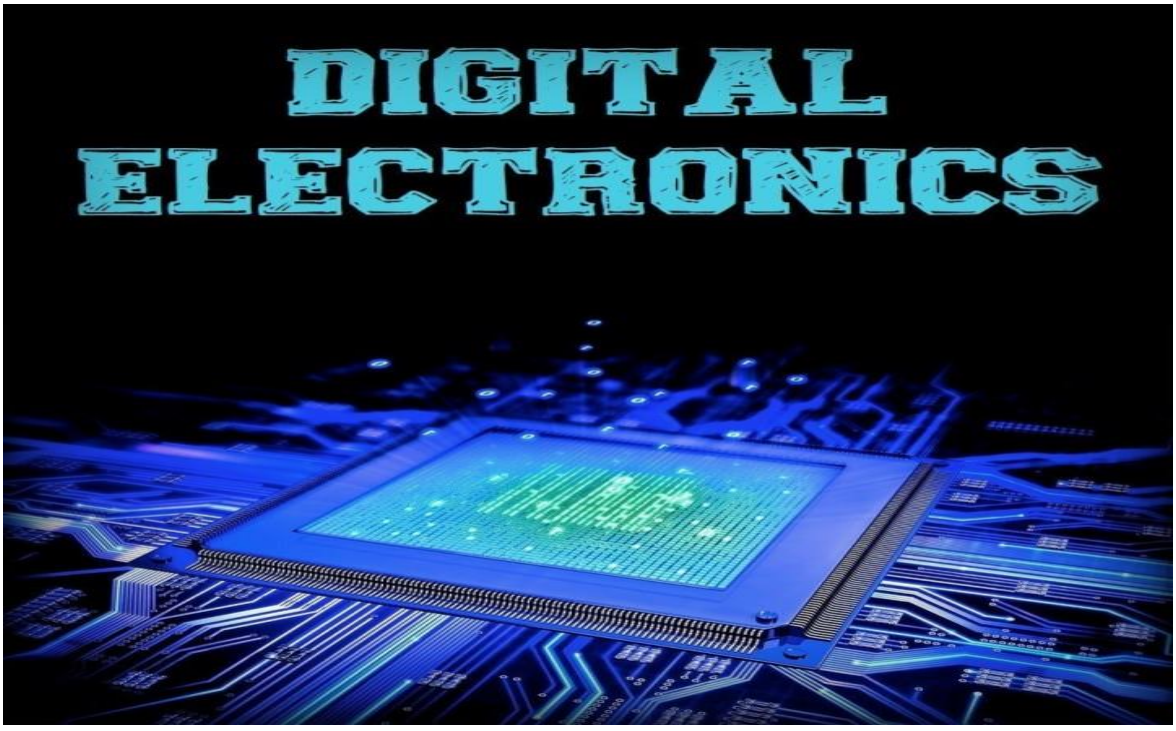
Abanob Evram

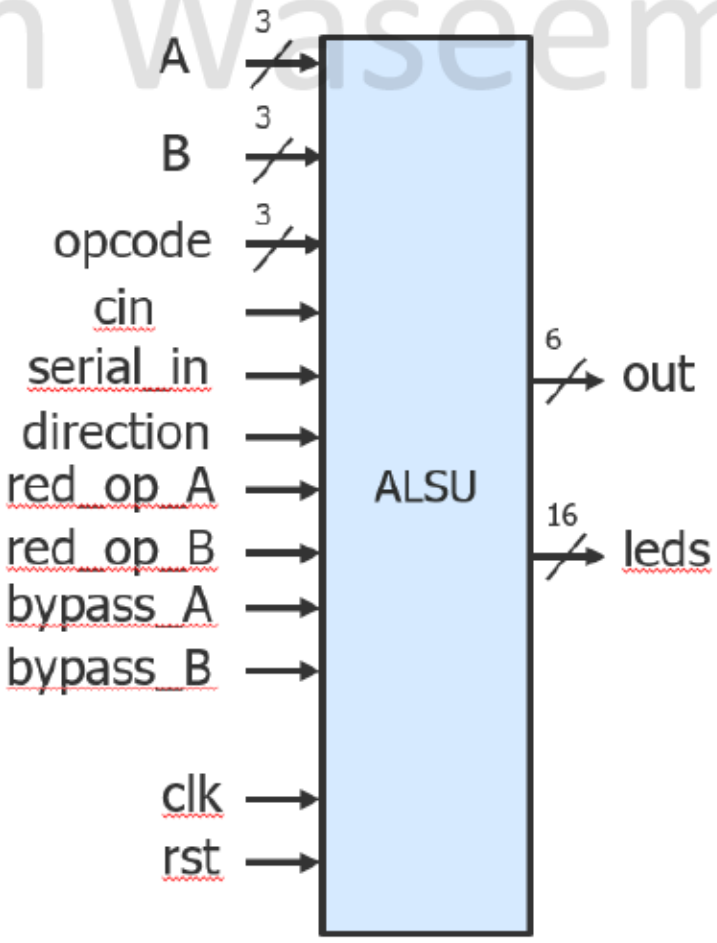
Assignmen4



[Q1]

1) ALSU is a logic unit that can perform logical, arithmetic, and shift operations on input ports

- Input ports A and B have various operations that can take place depending on the value of the opcode.
- Each input bit except for the clk and rst will be sampled at the rising edge before any processing so a D-FF is expected for each input bit at the design entry.
- The output of the ALSU is registered and is available at the rising edge of the clock.



Inputs

Each input bit except for the clk and rst will have a DFF in front of its port. Any processing will take place from the DFF output.

Input	Width	Description
clk	1	Input clock
rst	1	Active high asynchronous reset
A	3	Input port A
B	3	Input port B
cin	1	Carry in bit, only valid to be used if the parameter FULL_ADDER is "ON"
serial_in	1	Serial in bit, used in shift operations only
red_op_A	1	When set to high, this indicates that reduction operation would be executed on A rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
red_op_B	1	When set to high, this indicates that reduction operation would be executed on B rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
opcode	3	Opcode has a separate table to describe the different operations executed
bypass_A	1	When set to high, this indicates that port A will be registered to the output ignoring the opcode operation
bypass_B	1	When set to high, this indicates that port B will be registered to the output ignoring the opcode operation
direction	1	The direction of the shift or rotation operation is left when this input is set to high; otherwise, it is right.

Outputs and parameters

Output	Width	Description
leds	16	When an invalid operation occurs, all bits blink (bits turn on and then off with each clock cycle). Blinking serves as a warning; otherwise, if a valid operation occurs, it is set to low.
out	6	Output of the ALSU

Parameter	Default value	Description
INPUT_PRIORITY	A	Priority is given to the port set by this parameter whenever there is a conflict. Conflicts can occur in two scenarios, red_op_A and red_op_B are both set to high or bypass_A and bypass_B are both set to high. Legal values for this parameter are A and B
FULL_ADDER	ON	When this parameter has value "ON" then cin input must be considered in the addition operation between A and B. Legal values for this parameter are ON and OFF

Opcodes & Handling invalid cases

Opcode	Operation
000	AND
001	XOR
010	Addition
011	Multiplication
100	Shift output by 1 bit
101	Rotate output by 1 bit
110	Invalid opcode
111	Invalid opcode

Invalid cases

1. Opcode bits are set to 110 or 111
2. red_op_A or red_op_B are set to high and the opcode is not AND or XOR operation

Output when invalid cases occurs

1. leds are blinking
2. out bits are set to low, but if the bypass_A or bypass_B are high then the output will take the value of A or B.

The design code:

```
module ALSU(A,B,opcode,cin,serial_in,direction,red_op_A,red_op_B,bypass_A,bypass_B,clk,rst,out,leds);
parameter INPUT_PRIORITY="A";//A OR B
parameter FULL_ADDER="ON";//ON OR OFF
input cin,serial_in,direction,red_op_A,red_op_B,bypass_A,bypass_B,clk,rst;
input [2:0] A,B,opcode;
output reg [5:0] out ;
output reg [15:0] leds;
reg [2:0] Q_A,Q_B,Q_opcode;
reg Q_cin,Q_serial_in,Q_direction,Q_red_op_A,Q_red_op_B,Q_bypass_A,Q_bypass_B;

   always @(posedge clk or posedge rst) begin
       if (rst) begin
           Q_A<=0;
           Q_B<=0;
           Q_opcode<=0;
           Q_cin<=0;
           Q_serial_in<=0;
           Q_direction<=0;
           Q_red_op_A<=0;
           Q_red_op_B<=0;
           Q_bypass_A<=0;
           Q_bypass_B<=0;
       end
       else begin
           Q_A<=A;
           Q_B<=B;
           Q_opcode<=opcode;
           Q_cin<=cin;
           Q_serial_in<=serial_in;
           Q_direction<=direction;
           Q_red_op_A<=red_op_A;
           Q_red_op_B<=red_op_B;
           Q_bypass_A<=bypass_A;
           Q_bypass_B<=bypass_B;
       end
   end

   always @(posedge clk or posedge rst) begin
       if(rst)
           out<=0;

       else begin
           if(Q_bypass_A&&Q_bypass_B) begin
               if(INPUT_PRIORITY=="B")
                   out<=Q_B;
               else //default : INPUT_PRIORITY = A
                   out<=Q_A;
           end
           else if(Q_bypass_A)
               out<=Q_A;
           else if(Q_bypass_B)
               out<=Q_B;

           else begin
               case(Q_opcode)
```

```
55                  case(Q_opcode)
56                      0: begin //Operation: AND
57                          if (Q_red_op_A&&Q_red_op_B) begin
58                              if (INPUT_PRIORITY=="B")
59                                  out<=&Q_B;
60                              else
61                                  out<=&Q_A;
62                          end
63                          else if (Q_red_op_A)
64                                  out<=&Q_A;
65                          else if (Q_red_op_B)
66                                  out<=&Q_B;
67                          else
68                                  out<=Q_A&Q_B;
69                      end
70                      1: begin //Operation: XOR
71                          if (Q_red_op_A&&Q_red_op_B) begin
72                              if (INPUT_PRIORITY=="B")
73                                  out<=^Q_B;
74                              else
75                                  out<=^Q_A;
76                          end
77                          else if (Q_red_op_A)
78                                  out<=^Q_A;
79                          else if (Q_red_op_B)
80                              out<=^Q_B;
81                          else
82                              out<=Q_A^Q_B;
83                      end
84                      2: begin //Operation: Addition
85                          if(Q_red_op_A||Q_red_op_B)
86                              out<=0;
87                          else if (FULL_ADDER=="OFF")
88                              out<=Q_A+Q_B;
89                          else //default : FULL_ADDER= ON
90                              out<=Q_A+Q_B+Q_cin;
91                      end
92                      3: begin //Operation: Multiplication
93                          if(Q_red_op_A||Q_red_op_B)
94                              out<=0;
95                          else
96                              out<=Q_A*Q_B;
97                      end
98                      4: begin //Operation: Shift output by 1 bit
99                          if(Q_red_op_A||Q_red_op_B)
100                             out<=0;
101                         else begin
102                             if (Q_direction) // if direction ==1 then shift left
103                                 out<={out[4:0],Q_serial_in};
104                             else //shift right
105                                 out<={Q_serial_in,out[5:1]};
106                         end
107                     end
108                     5: begin //Operation: Rotate output by 1 bit
109                         if(Q_red_op_A||Q_red_op_B)
```

```
                        if(Q_red_op_A||Q_red_op_B)
                            out<=0;
                        else if (FULL_ADDER=="OFF")
                            out<=Q_A+Q_B;
                        else //default : FULL_ADDER= ON
                            out<=Q_A+Q_B+Q_cin;
                    end
                    3: begin //Operation: Multiplication
                        if(Q_red_op_A||Q_red_op_B)
                            out<=0;
                        else
                            out<=Q_A*Q_B;
                    end
                    4: begin //Operation: Shift output by 1 bit
                        if(Q_red_op_A||Q_red_op_B)
                            out<=0;
                        else begin
                            if (Q_direction) // if direction ==1 then shift left
                                out<={out[4:0],Q_serial_in};
                            else //shift right
                                out<={Q_serial_in,out[5:1]};
                        end
                    end
                    5: begin //Operation: Rotate output by 1 bit
                        if(Q_red_op_A||Q_red_op_B)
                            out<=0;
                        else begin
                            if (Q_direction) // if direction ==1 then Rotate left
                                out<={out[4:0],out[5]};
                            else //Rotate right
                                out<={out[0],out[5:1]};
                        end
                    end
                    default : out<=0;
                endcase
            end
        end
    end
    always @(posedge clk or posedge rst) begin
        if (rst)
            leds<=0;
        else if (Q_opcode==0||Q_opcode==1)
            leds<=0;
        else if (Q_opcode==2||Q_opcode==3||Q_opcode==4||Q_opcode==5) begin
            if(Q_red_op_A||Q_red_op_B) begin
                leds<=~leds;
            end
            else
                leds<=0;
        end
        else begin
            leds<=~leds;
        end
    end
endmodule
```

The testbench code:

```
module ALSU_tb();
parameter INPUT_PRIORITY_TB="A";//A OR B
parameter FULL_ADDER_TB="ON";//ON OR OFF
reg cin_tb,serial_in_tb,direction_tb,red_op_A_tb,red_op_B_tb,bypass_A_tb,bypass_B_tb,clk_tb,rst_tb;
reg [2:0] A_tb,B_tb,opcode_tb;
wire [5:0] out_dut;
wire [15:0] leds_dut;

ALSU #(INPUT_PRIORITY_TB,FULL_ADDER_TB) dut(A_tb,B_tb,opcode_tb,cin_tb,serial_in_tb,direction_tb,red_op_A_tb,red_op_B_tb,bypass_A_tb,bypass_B_tb,clk_tb,rst_tb,out_dut,leds_dut);
integer i;
   initial begin
       clk_tb=0;
       forever
           #1 clk_tb=~clk_tb;
   end
   initial begin
       rst_tb=1;cin_tb=0;serial_in_tb=0;direction_tb=0;red_op_A_tb=0;red_op_B_tb=0;bypass_A_tb=0;bypass_B_tb=0; // test if rst = 1
       for(i=0;i<30;i=i+1) begin
           A_tb=$random;
           B_tb=$random;
           opcode_tb=$random;
           @(negedge clk_tb);
       end
       rst_tb=0;bypass_A_tb=1;bypass_B_tb=1; //test if bypass A&B =1
       for(i=0;i<30;i=i+1) begin
           A_tb=$random;
           B_tb=$random;
           opcode_tb=$random;
           @(negedge clk_tb);
       end
       bypass_B_tb=0; //test if bypass A=1
       for(i=0;i<30;i=i+1) begin
           A_tb=$random;
           B_tb=$random;
           opcode_tb=$random;
           @(negedge clk_tb);
       end
       bypass_B_tb=1;bypass_A_tb=0; //test if bypass B=1
       for(i=0;i<30;i=i+1) begin
           A_tb=$random;
           B_tb=$random;
           opcode_tb=$random;
           @(negedge clk_tb);
       end

       bypass_B_tb=0;red_op_A_tb=1;red_op_B_tb=1;opcode_tb=0; //test AND if red_op_A&B=1 :(INPUT_PRIORITY=A)
       for(i=0;i<30;i=i+1) begin
           A_tb=$random;
           B_tb=$random;
           @(negedge clk_tb);
       end
       red_op_A_tb=0;//test AND if red_op_B=1
       for(i=0;i<30;i=i+1) begin
           A_tb=$random;
           B_tb=$random;
```

```
        red_op_A_tb=0;//test AND if red_op_B=1
        for(i=0;i<30;i=i+1) begin
            A_tb=$random;
            B_tb=$random;
            @(negedge clk_tb);
        end
        red_op_B_tb=0;//test AND if red_op_B=0&red_op_B=0
        for(i=0;i<30;i=i+1) begin
            A_tb=$random;
            B_tb=$random;
            @(negedge clk_tb);
        end

        red_op_A_tb=1;red_op_B_tb=1;opcode_tb=1; //test XOR if red_op_A&B=1 :(INPUT_PRIORITY=A)
        for(i=0;i<30;i=i+1) begin
            A_tb=$random;
            B_tb=$random;
            @(negedge clk_tb);
        end
        red_op_A_tb=0;//test XOR if red_op_B=1
        for(i=0;i<30;i=i+1) begin
            A_tb=$random;
            B_tb=$random;
            @(negedge clk_tb);
        end
        red_op_B_tb=0;//test XOR if red_op_B=0&red_op_B=0
        for(i=0;i<30;i=i+1) begin
            A_tb=$random;
            B_tb=$random;
            @(negedge clk_tb);
        end

        opcode_tb=2; //test addition
        for(i=0;i<30;i=i+1) begin
            A_tb=$random;
            B_tb=$random;
            cin_tb=$random;
            @(negedge clk_tb);
        end

        opcode_tb=3; //test multiplication
        for(i=0;i<30;i=i+1) begin
            A_tb=$random;
            B_tb=$random;
            @(negedge clk_tb);
        end

        opcode_tb=4;direction_tb=1;//test shift left
        for(i=0;i<30;i=i+1) begin
            serial_in_tb=$random;
            @(negedge clk_tb);
        end
        direction_tb=0; //test shift right
        for(i=0;i<30;i=i+1) begin
            serial_in_tb=$random;
            @(negedge clk_tb);
```

```
76          end
77          red_op_B_tb=0;//test XOR if red_op_B=0&red_op_B=0
78          for(i=0;i<30;i=i+1) begin
79              A_tb=$random;
80              B_tb=$random;
81              @(negedge clk_tb);
82          end
83
84          opcode_tb=2; //test addition
85          for(i=0;i<30;i=i+1) begin
86              A_tb=$random;
87              B_tb=$random;
88              cin_tb=$random;
89              @(negedge clk_tb);
90          end
91
92          opcode_tb=3; //test multiplication
93          for(i=0;i<30;i=i+1) begin
94              A_tb=$random;
95              B_tb=$random;
96              @(negedge clk_tb);
97          end
98
99          opcode_tb=4;direction_tb=1;//test shift left
100         for(i=0;i<30;i=i+1) begin
101             serial_in_tb=$random;
102             @(negedge clk_tb);
103         end
104         direction_tb=0; //test shift right
105         for(i=0;i<30;i=i+1) begin
106             serial_in_tb=$random;
107             @(negedge clk_tb);
108         end
109
110         opcode_tb=5;//test rotate right
111         for(i=0;i<30;i=i+1) begin
112             serial_in_tb=$random;
113             @(negedge clk_tb);
114         end
115         direction_tb=1;//test rotate left
116         for(i=0;i<30;i=i+1) begin
117             serial_in_tb=$random;
118             @(negedge clk_tb);
119         end
120         red_op_A_tb=1;red_op_B_tb=1;// test some invalid cases
121         for(i=0;i<100;i=i+1) begin
122             A_tb=$random;
123             B_tb=$random;
124             opcode_tb=$urandom_range(2,7);
125             @(negedge clk_tb);
126         end
127     $stop;
128     end
129 endmodule
```

The do file code:

```
vlib work

vlog ALSU.v ALSU_tb.v

vsim -voptargs=+acc ALSU_tb

add wave *

run -all

#quit -sim
```

The constraint code:

```
## Clock signal
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports clk]
create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports clk]


## Switches
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {opcode[0]}]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {opcode[1]}]
set_property -dict {PACKAGE_PIN W16 IOSTANDARD LVCMOS33} [get_ports {opcode[2]}]
set_property -dict {PACKAGE_PIN W17 IOSTANDARD LVCMOS33} [get_ports {A[0]}]
set_property -dict {PACKAGE_PIN W15 IOSTANDARD LVCMOS33} [get_ports {A[1]}]
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {A[2]}]
set_property -dict {PACKAGE_PIN W14 IOSTANDARD LVCMOS33} [get_ports {B[0]}]
set_property -dict {PACKAGE_PIN W13 IOSTANDARD LVCMOS33} [get_ports {B[1]}]
set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {B[2]}]
set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVCMOS33} [get_ports cin]
set_property -dict {PACKAGE_PIN T2 IOSTANDARD LVCMOS33} [get_ports red_op_A]
set_property -dict {PACKAGE_PIN R3 IOSTANDARD LVCMOS33} [get_ports red_op_B]
set_property -dict {PACKAGE_PIN W2 IOSTANDARD LVCMOS33} [get_ports bypass_A]
set_property -dict {PACKAGE_PIN U1 IOSTANDARD LVCMOS33} [get_ports bypass_B]
set_property -dict {PACKAGE_PIN T1 IOSTANDARD LVCMOS33} [get_ports direction]
set_property -dict {PACKAGE_PIN R2 IOSTANDARD LVCMOS33} [get_ports serial_in]


## LEDs
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {leds[0]}]
set_property -dict {PACKAGE_PIN E19 IOSTANDARD LVCMOS33} [get_ports {leds[1]}]
set_property -dict {PACKAGE_PIN U19 IOSTANDARD LVCMOS33} [get_ports {leds[2]}]
set_property -dict {PACKAGE_PIN V19 IOSTANDARD LVCMOS33} [get_ports {leds[3]}]
set_property -dict {PACKAGE_PIN W18 IOSTANDARD LVCMOS33} [get_ports {leds[4]}]
set_property -dict {PACKAGE_PIN U15 IOSTANDARD LVCMOS33} [get_ports {leds[5]}]
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {leds[6]}]
set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {leds[7]}]
set_property -dict {PACKAGE_PIN V13 IOSTANDARD LVCMOS33} [get_ports {leds[8]}]
set_property -dict {PACKAGE_PIN V3 IOSTANDARD LVCMOS33} [get_ports {leds[9]}]
set_property -dict {PACKAGE_PIN W3 IOSTANDARD LVCMOS33} [get_ports {leds[10]}]
set_property -dict {PACKAGE_PIN U3 IOSTANDARD LVCMOS33} [get_ports {leds[11]}]
set_property -dict {PACKAGE_PIN P3 IOSTANDARD LVCMOS33} [get_ports {leds[12]}]
set_property -dict {PACKAGE_PIN N3 IOSTANDARD LVCMOS33} [get_ports {leds[13]}]
set_property -dict {PACKAGE_PIN P1 IOSTANDARD LVCMOS33} [get_ports {leds[14]}]
set_property -dict {PACKAGE_PIN L1 IOSTANDARD LVCMOS33} [get_ports {leds[15]}]
```

```
## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]

create_debug_core u_ila_0 ila
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
set_property port_width 3 [get_debug_ports u_ila_0/probe0]
connect_debug_port u_ila_0/probe0 [get_nets [list {B_IBUF[0]} {B_IBUF[1]} {B_IBUF[2]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
set_property port_width 16 [get_debug_ports u_ila_0/probe1]
connect_debug_port u_ila_0/probe1 [get_nets [list {leds_OBUF[0]} {leds_OBUF[1]} {leds_OBUF[2]} {leds_OBUF[3]} {leds_OBUF[4]} {leds_OBUF[5]} {leds_OBUF[6]}
{leds_OBUF[7]} {leds_OBUF[8]} {leds_OBUF[9]} {leds_OBUF[10]} {leds_OBUF[11]} {leds_OBUF[12]} {leds_OBUF[13]} {leds_OBUF[14]} {leds_OBUF[15]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
set_property port_width 3 [get_debug_ports u_ila_0/probe2]
connect_debug_port u_ila_0/probe2 [get_nets [list {opcode_IBUF[0]} {opcode_IBUF[1]} {opcode_IBUF[2]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
set_property port_width 6 [get_debug_ports u_ila_0/probe3]
connect_debug_port u_ila_0/probe3 [get_nets [list {out_OBUF[0]} {out_OBUF[1]} {out_OBUF[2]} {out_OBUF[3]} {out_OBUF[4]} {out_OBUF[5]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
set_property port_width 3 [get_debug_ports u_ila_0/probe4]
connect_debug_port u_ila_0/probe4 [get_nets [list {A_IBUF[0]} {A_IBUF[1]} {A_IBUF[2]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe5]
set_property port_width 1 [get_debug_ports u_ila_0/probe5]
connect_debug_port u_ila_0/probe5 [get_nets [list bypass_A_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe6]
set_property port_width 1 [get_debug_ports u_ila_0/probe6]
connect_debug_port u_ila_0/probe6 [get_nets [list bypass_B_IBUF]]
```

```
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
set_property port_width 6 [get_debug_ports u_ila_0/probe3]
connect_debug_port u_ila_0/probe3 [get_nets [list {out_OBUF[0]} {out_OBUF[1]} {out_OBUF[2]} {out_OBUF[3]} {out_OBUF[4]} {out_OBUF[5]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
set_property port_width 3 [get_debug_ports u_ila_0/probe4]
connect_debug_port u_ila_0/probe4 [get_nets [list {A_IBUF[0]} {A_IBUF[1]} {A_IBUF[2]}]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe5]
set_property port_width 1 [get_debug_ports u_ila_0/probe5]
connect_debug_port u_ila_0/probe5 [get_nets [list bypass_A_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe6]
set_property port_width 1 [get_debug_ports u_ila_0/probe6]
connect_debug_port u_ila_0/probe6 [get_nets [list bypass_B_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe7]
set_property port_width 1 [get_debug_ports u_ila_0/probe7]
connect_debug_port u_ila_0/probe7 [get_nets [list cin_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe8]
set_property port_width 1 [get_debug_ports u_ila_0/probe8]
connect_debug_port u_ila_0/probe8 [get_nets [list clk_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe9]
set_property port_width 1 [get_debug_ports u_ila_0/probe9]
connect_debug_port u_ila_0/probe9 [get_nets [list direction_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe10]
set_property port_width 1 [get_debug_ports u_ila_0/probe10]
connect_debug_port u_ila_0/probe10 [get_nets [list red_op_A_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe11]
set_property port_width 1 [get_debug_ports u_ila_0/probe11]
connect_debug_port u_ila_0/probe11 [get_nets [list red_op_B_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe12]
set_property port_width 1 [get_debug_ports u_ila_0/probe12]
connect_debug_port u_ila_0/probe12 [get_nets [list rst_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe13]
set_property port_width 1 [get_debug_ports u_ila_0/probe13]
connect_debug_port u_ila_0/probe13 [get_nets [list serial_in_IBUF]]
set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]

```

```
##Buttons
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports rst]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports btnU]
#set_property -dict { PACKAGE_PIN W19   IOSTANDARD LVCMOS33 } [get_ports btnL]
#set_property -dict { PACKAGE_PIN T17   IOSTANDARD LVCMOS33 } [get_ports btnR]
#set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports btnD]
```

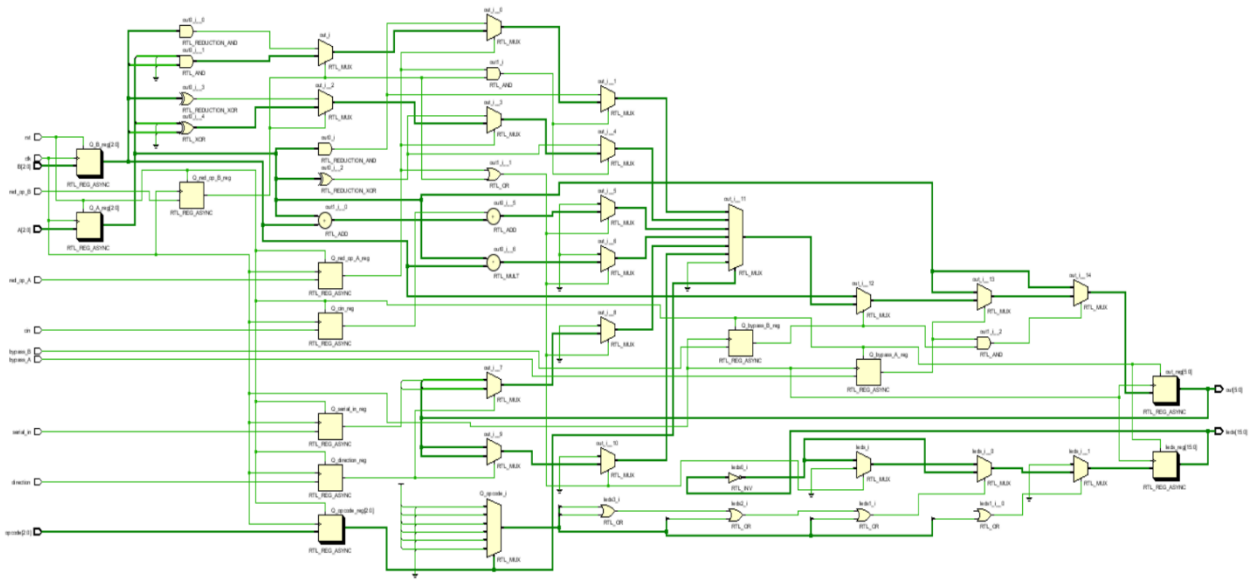
1-Elaboration (messages_tab)

Tcl Console | Messages | Log | Reports | Design Runs

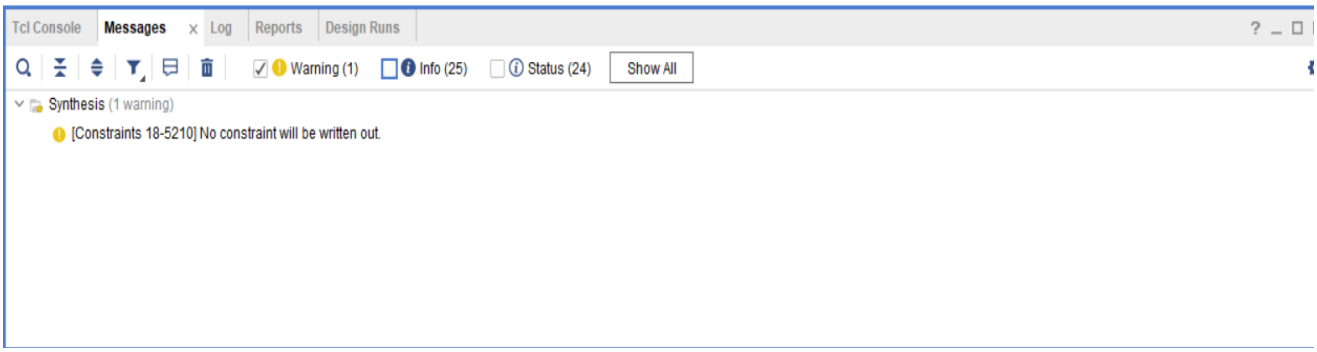
Info (8) Status (11) Show All

- Vivado Commands (3 infos)
  - General Messages (3 infos)
    - [IP_Flow 19-234] Refreshing IP repositories
    - [IP_Flow 19-1704] No user IP repositories specified
    - [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/Vivado/Vivado/2018.2/data/ip'.
- Elaborated Design (5 infos)
  - General Messages (5 infos)
    - [Synth 8-6157] synthesizing module 'ALSU' [ALSU.v:1]
    - [Synth 8-6155] done synthesizing module 'ALSU' (1#1) [ALSU.v:1]
    - [Project 1-570] Preparing netlist for logic optimization
    - [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
    - [Project 1-111] Unisim Transformation Summary:
      No Unisim elements were transformed.

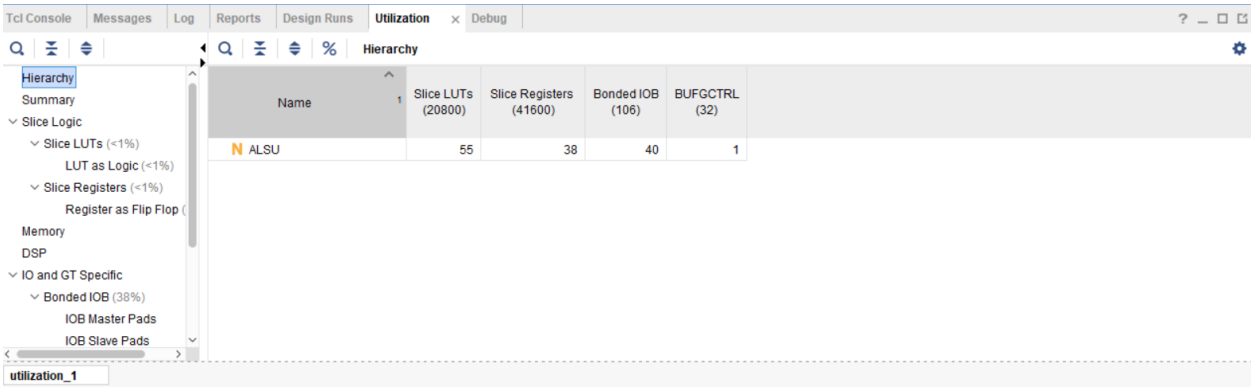
2-Elaboration(Schematic)



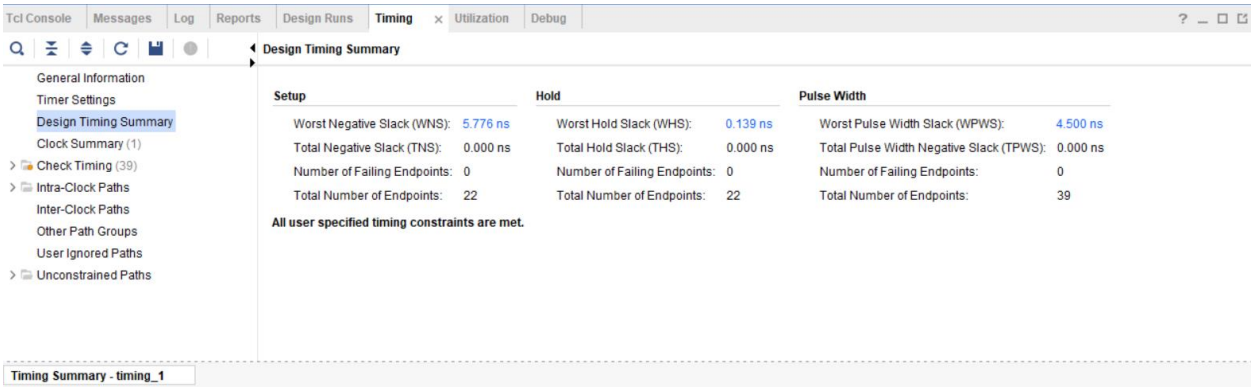
3-synthesis(messages_tab)



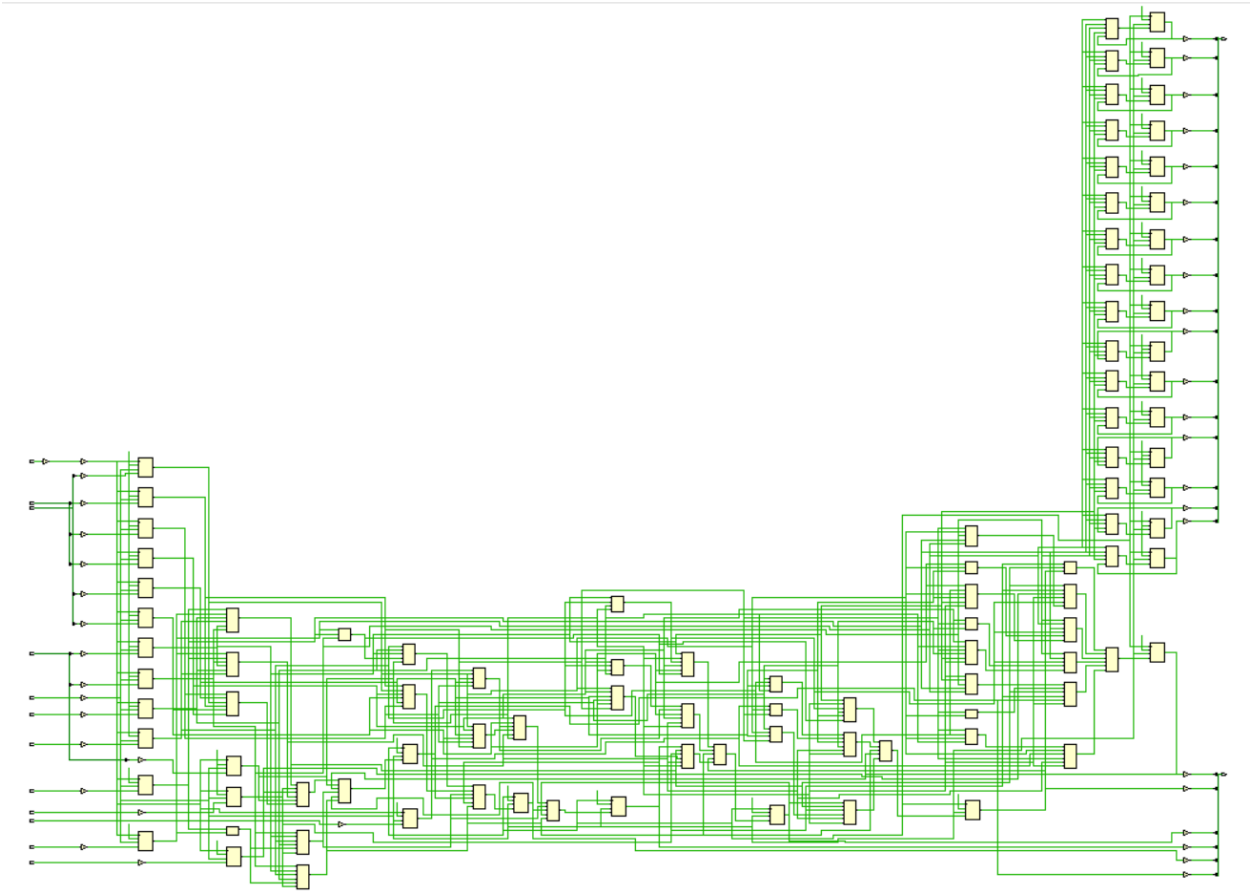
4-synthesis(Utilization_report)



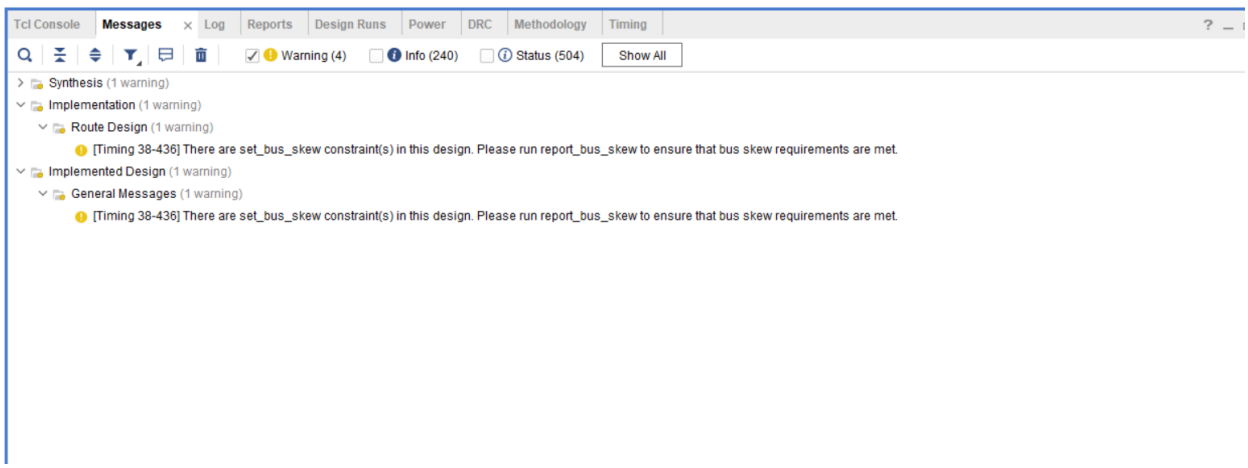
5-synthesis(Timing_report)



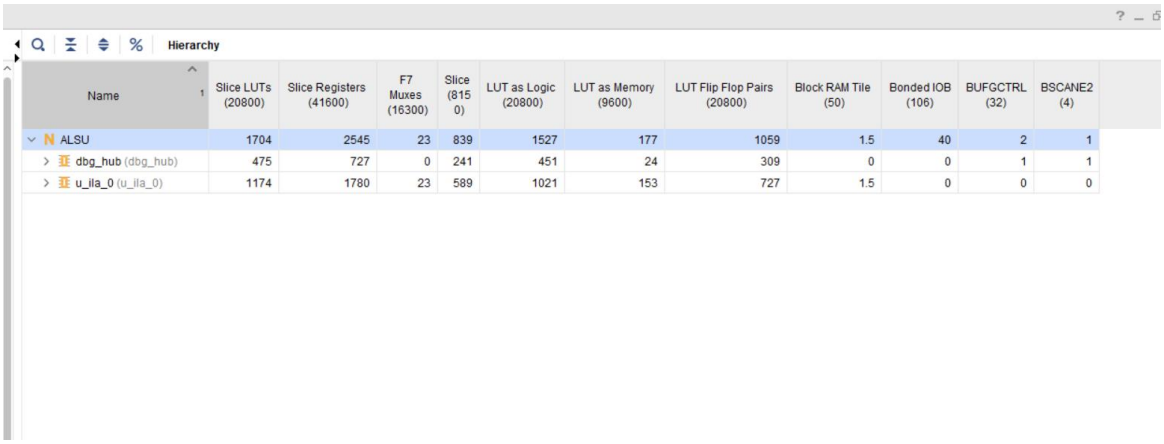
6-synthesis(Schematic)



7-implmentation(messages_tab)



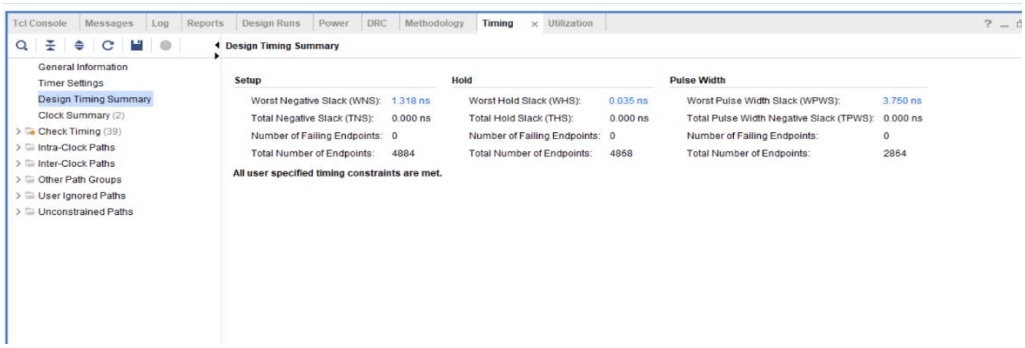
8-Implmentation(Utilization_report)



Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	LUT Flip Flop Pairs (20800)	Block RAM Tile (50)	Bonded IOB (106)	BUFGCTRL (32)	BSCANE2 (4)
ALSU	1704	2545	23	839	1527	177	1059	1.5	40	2	1
dbg_hub (dbg_hub)	475	727	0	241	451	24	309	0	0	1	1
u_ila_0 (u_ila_0)	1174	1780	23	589	1021	153	727	1.5	0	0	0

9-Implmentation(Timing_report)



Tcl Console | Messages | Log | Reports | Design Runs | Power | DRC | Methodology | Timing | Utilization

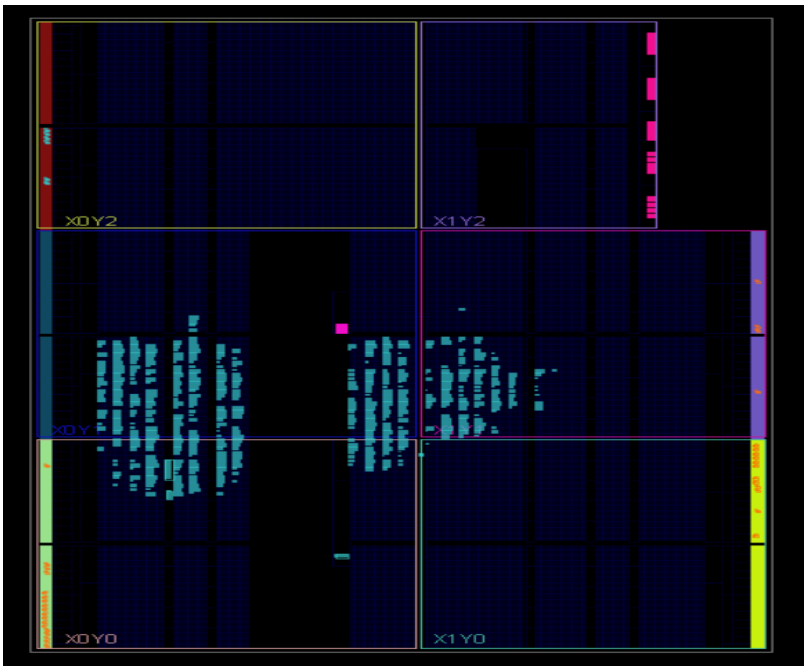
- General Information
- Timer Settings
- Design Timing Summary
- Clock Summary (2)
- Check Timing (39)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths

Design Timing Summary

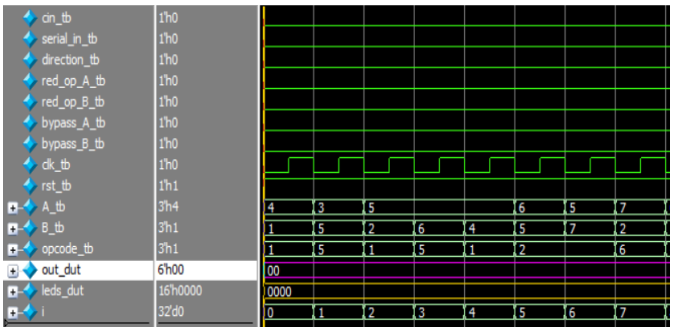
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.318 ns	Worst Hold Slack (WHS):	0.035 ns	Worst Pulse Width Slack (WPWS):	3.750 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	4884	Total Number of Endpoints:	4868	Total Number of Endpoints:	2864

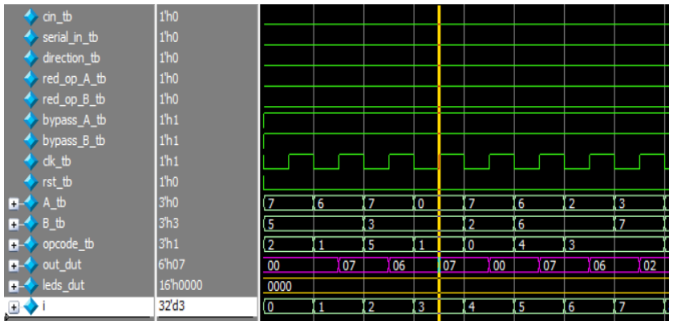
All user specified timing constraints are met.

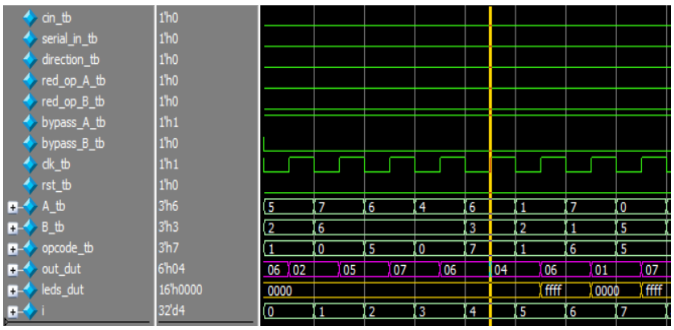
10-Implmentation(device)

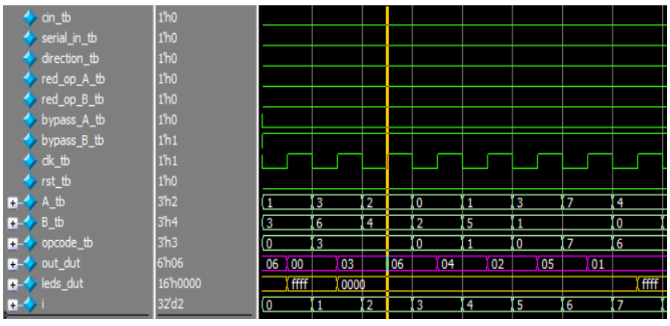


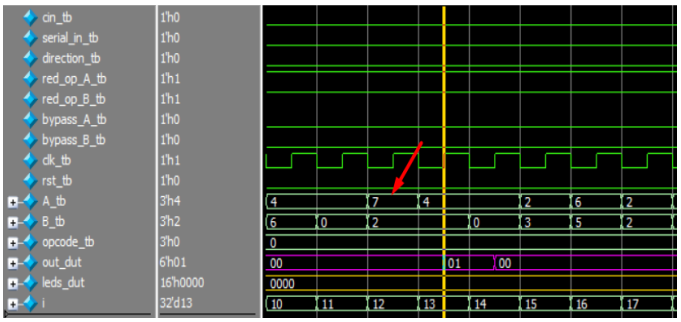
Some snippets of waves:

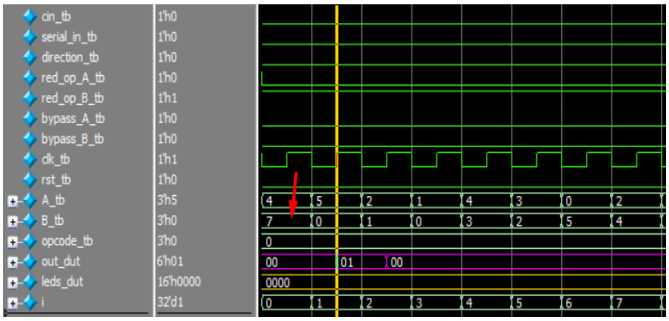


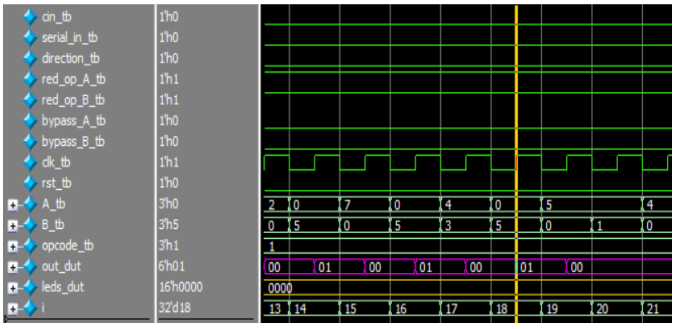


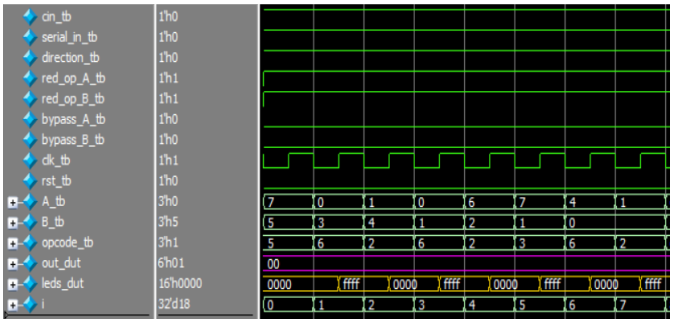






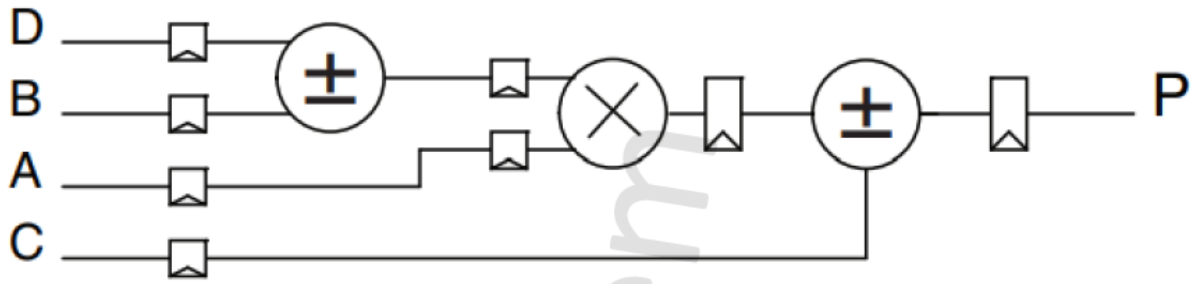






[Q2]

2) Design the following simplified version of the DSP block DSP48A1 from Xilinx Spartan-6 FPGA. Use a directed testbench to simplify the verification. Check the waveform to make sure that the operations are done correctly at every pipeline stage.



Port	Type	Width
A	Input	18
B	Input	18
C	Input	48
D	Input	18
clk	Input	1
rst_n (sync active low)	Input	1
P	Output	48

Parameters:

- OPERATION: take 2 values either "ADD" or "SUBTRACT", Default value "ADD"
  - When subtracting use "D - B" and "multiplier_out – C". multiplier_out is an internal signal

The design (first code):

```
module Simple_DSP(A,B,C,D,clk,rst_n,P);
parameter OPERATION ="ADD";//ADD OR SUBTRACT
input [17:0] A,B,D;
input [47:0] C;
input clk,rst_n;
output  [47:0] P;
reg [17:0]  Adder1 ;
reg [47:0]  Adder2 ;
reg [47:0] mult;
wire [17:0] Q_D,Q_B,Q_A1,Q_A2,Q_Adder1;
wire [47:0] Q_C,Q_mult;

D_Flipflop     #(18)   d0(D,rst_n,clk,Q_D);
D_Flipflop     #(18)   d1(B,rst_n,clk,Q_B);
D_Flipflop     #(18)   d2(A,rst_n,clk,Q_A1);
D_Flipflop     #(48)   d3(C,rst_n,clk,Q_C);

D_Flipflop     #(18)   d4(Adder1,rst_n,clk,Q_Adder1);
D_Flipflop     #(18)   d5(Q_A1,rst_n,clk,Q_A2);
D_Flipflop     #(48)   d6(mult,rst_n,clk,Q_mult);
D_Flipflop     #(48)   d7(Adder2,rst_n,clk,P);

always @(*) begin
   if (OPERATION=="SUBTRACT")
       Adder1= Q_D - Q_B;

   else
       Adder1= Q_D + Q_B;

   mult = Q_A2*Q_Adder1;

   if (OPERATION=="SUBTRACT")
       Adder2= Q_mult - Q_C;

   else
       Adder2= Q_mult + Q_C;
end
endmodule
```

```
module D_Flipflop(d,rstn,clk,q);
parameter WIDTH=1;
input [WIDTH-1:0] d;
input rstn,clk;
output reg [WIDTH-1:0] q;
    always @(posedge clk) begin
        if (~rstn)
            q<=0;
        else
            q<=d;
    end
endmodule
```

The design (second code):

```
module Simple_DSP2(A,B,C,D,clk,rst_n,p);
 parameter OPERATION ="ADD";//ADD OR SUBTRACT
 input [17:0] A,B,D;
 input [47:0] C;
 input clk,rst_n;
 output reg [47:0] p;
 reg [17:0] Q_D,Q_B,Q_A1,Q_A2,Q_Adder1;
 reg [47:0] Q_C,Q_mult;
 always @(posedge clk ) begin
    if (~rst_n) begin
        Q_D<=0;
        Q_B<=0;
        Q_A1<=0;
        Q_A2<=0;
        Q_Adder1<=0;
        Q_C<=0;
        Q_mult<=0;
        p<=0;
    end
    else begin
        Q_D<=D;
        Q_B<=B;
        Q_C<=C;
        Q_A1<=A;
        Q_A2<=Q_A1;
        Q_mult<=Q_A2*Q_Adder1;
        if(OPERATION=="SUBTRACT") begin
            Q_Adder1<= Q_D - Q_B;
            p<= Q_mult - Q_C;
        end
        else begin
            Q_Adder1<= Q_D + Q_B;
            p<= Q_mult + Q_C;
        end
    end
 end
endmodule
```

The testbench code:

```
module DSP_block_tb ();
 parameter OPERATION_tb="ADD";
 reg [17:0] D_tb,B_tb,A_tb;
 reg  [47:0]C_tb;
 reg clk_tb,rst_n_tb;
 wire [47:0]P_dut;
 Simple_DSP #(OPERATION_tb) DUT(A_tb,B_tb,C_tb,D_tb,clk_tb,rst_n_tb,P_dut);
 initial begin
    clk_tb=0;
    forever
        #1 clk_tb=~clk_tb;
 end
 initial begin
    rst_n_tb=0;
    D_tb=0;B_tb=0;A_tb=0;C_tb=0;
    @(negedge clk_tb );

    rst_n_tb=1;D_tb=5;B_tb=5;A_tb=10;C_tb=10;//p=105
    @(negedge clk_tb );

    D_tb=5;B_tb=5;A_tb=5;C_tb=5;//p=55
    @(negedge clk_tb );

    D_tb=15;B_tb=15;A_tb=5;C_tb=5;//p=155
    @(negedge clk_tb );

    D_tb=20;B_tb=20;A_tb=5;C_tb=5;//p=205
    @(negedge clk_tb );

    D_tb=2;B_tb=5;A_tb=5;C_tb=5;//p=40
    @(negedge clk_tb );

    D_tb=3;B_tb=5;A_tb=5;C_tb=5;//p=45
    @(negedge clk_tb );

    D_tb=4;B_tb=5;A_tb=5;C_tb=5;//p=50
    @(negedge clk_tb );

    D_tb=15;B_tb=7;A_tb=2;C_tb=5;//p=49
    @(negedge clk_tb );


    D_tb=18;B_tb=2;A_tb=4;C_tb=5;//p=85
    @(negedge clk_tb );

    D_tb=8;B_tb=2;A_tb=9;C_tb=6;//p=96
    @(negedge clk_tb );
    D_tb=16;B_tb=1;A_tb=4;C_tb=7;//p=75
    repeat (5)
    @(negedge clk_tb );
    $stop;
    end
endmodule
```

The do file code:

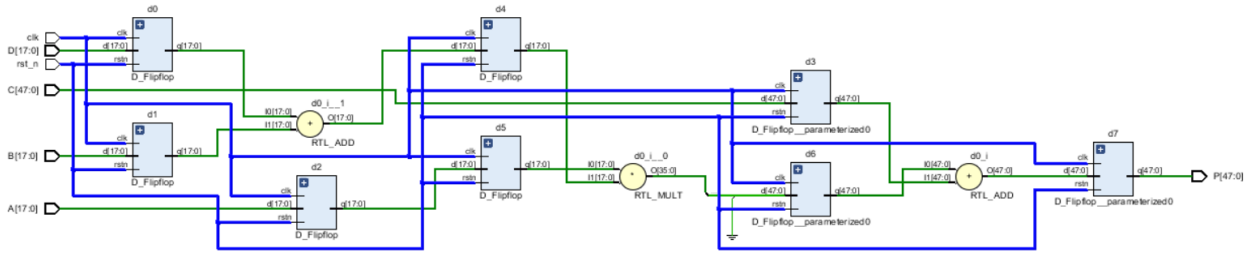
```
vlib work

vlog DSP.v D_Flipflop.v DSP_BLOCK_tb.v DSP_block_tb.v

vsim -voptargs=+acc DSP_block_tb

add wave *

run -all

#quit -sim
```

The constraint code:

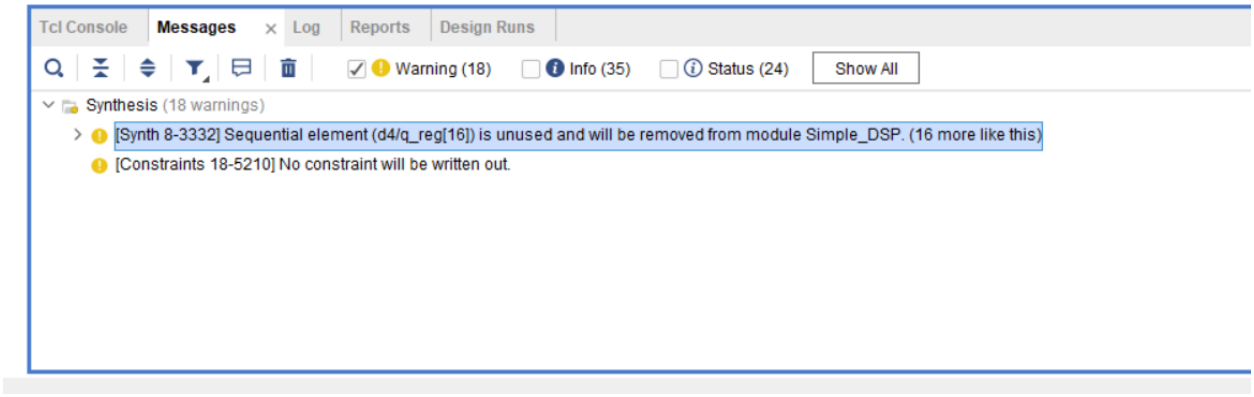
```
## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]

create_debug_core u_ila_0 ila
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets [list clk_IBUF_BUFG]]
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
set_property port_width 18 [get_debug_ports u_ila_0/probe0]
connect_debug_port u_ila_0/probe0 [get_nets [list {B_IBUF[0]} {B_IBUF[1]} {B_IBUF[2]} {B_IBUF[3]} {B_IBUF[4]} {B_IBUF[5]} {B_IBUF
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
set_property port_width 48 [get_debug_ports u_ila_0/probe1]
connect_debug_port u_ila_0/probe1 [get_nets [list {P_OBUF[0]} {P_OBUF[1]} {P_OBUF[2]} {P_OBUF[3]} {P_OBUF[4]} {P_OBUF[5]} {P_OBUF
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe2]
set_property port_width 18 [get_debug_ports u_ila_0/probe2]
connect_debug_port u_ila_0/probe2 [get_nets [list {A_IBUF[0]} {A_IBUF[1]} {A_IBUF[2]} {A_IBUF[3]} {A_IBUF[4]} {A_IBUF[5]} {A_IBUF
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe3]
set_property port_width 48 [get_debug_ports u_ila_0/probe3]
connect_debug_port u_ila_0/probe3 [get_nets [list {C_IBUF[0]} {C_IBUF[1]} {C_IBUF[2]} {C_IBUF[3]} {C_IBUF[4]} {C_IBUF[5]} {C_IBUF
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe4]
set_property port_width 18 [get_debug_ports u_ila_0/probe4]
connect_debug_port u_ila_0/probe4 [get_nets [list {D_IBUF[0]} {D_IBUF[1]} {D_IBUF[2]} {D_IBUF[3]} {D_IBUF[4]} {D_IBUF[5]} {D_IBUF
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe5]
set_property port_width 1 [get_debug_ports u_ila_0/probe5]
connect_debug_port u_ila_0/probe5 [get_nets [list clk_IBUF]]
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe6]
set_property port_width 1 [get_debug_ports u_ila_0/probe6]
connect_debug_port u_ila_0/probe6 [get_nets [list rst_n_IBUF]]
set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
```

1-Elaboration (messages_tab)

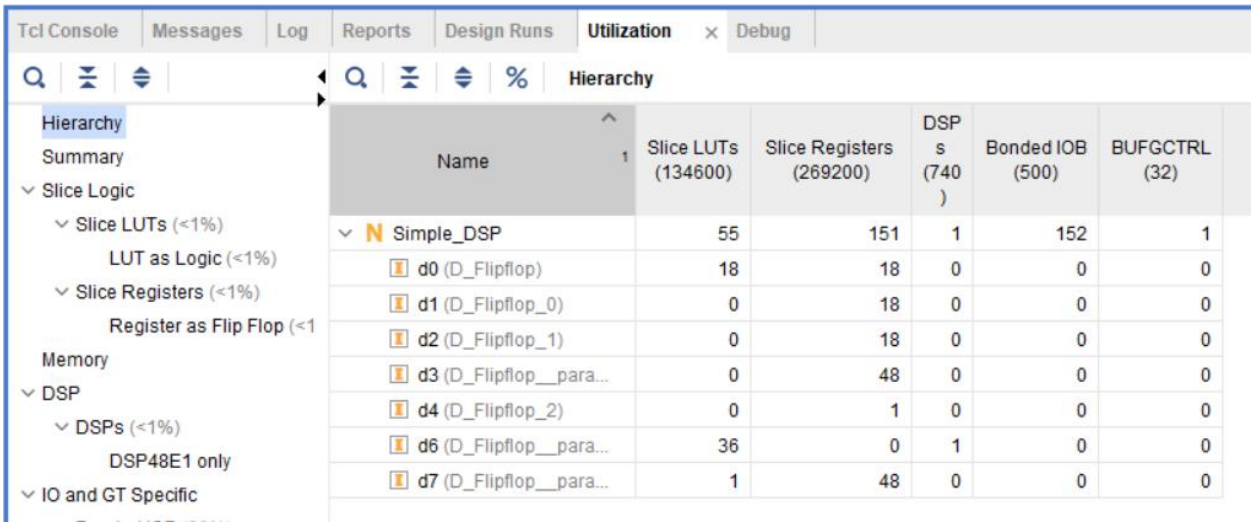
2-Elaboration(Schematic)



3-synthesis(messages_tab)

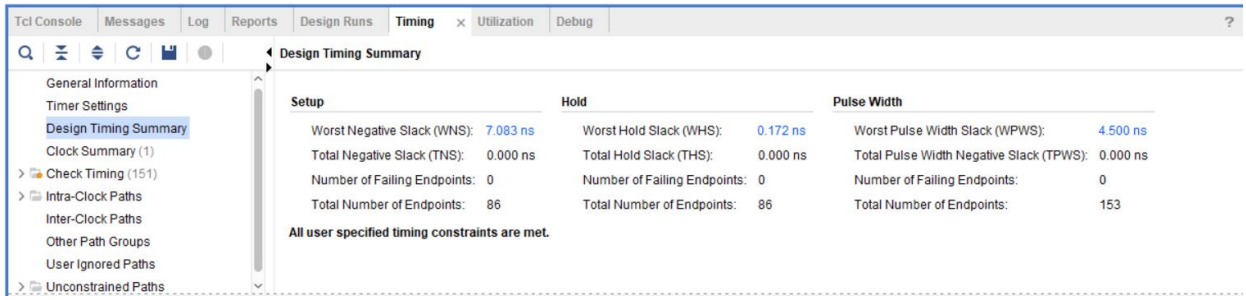


4-synthesis(Utilization_report)

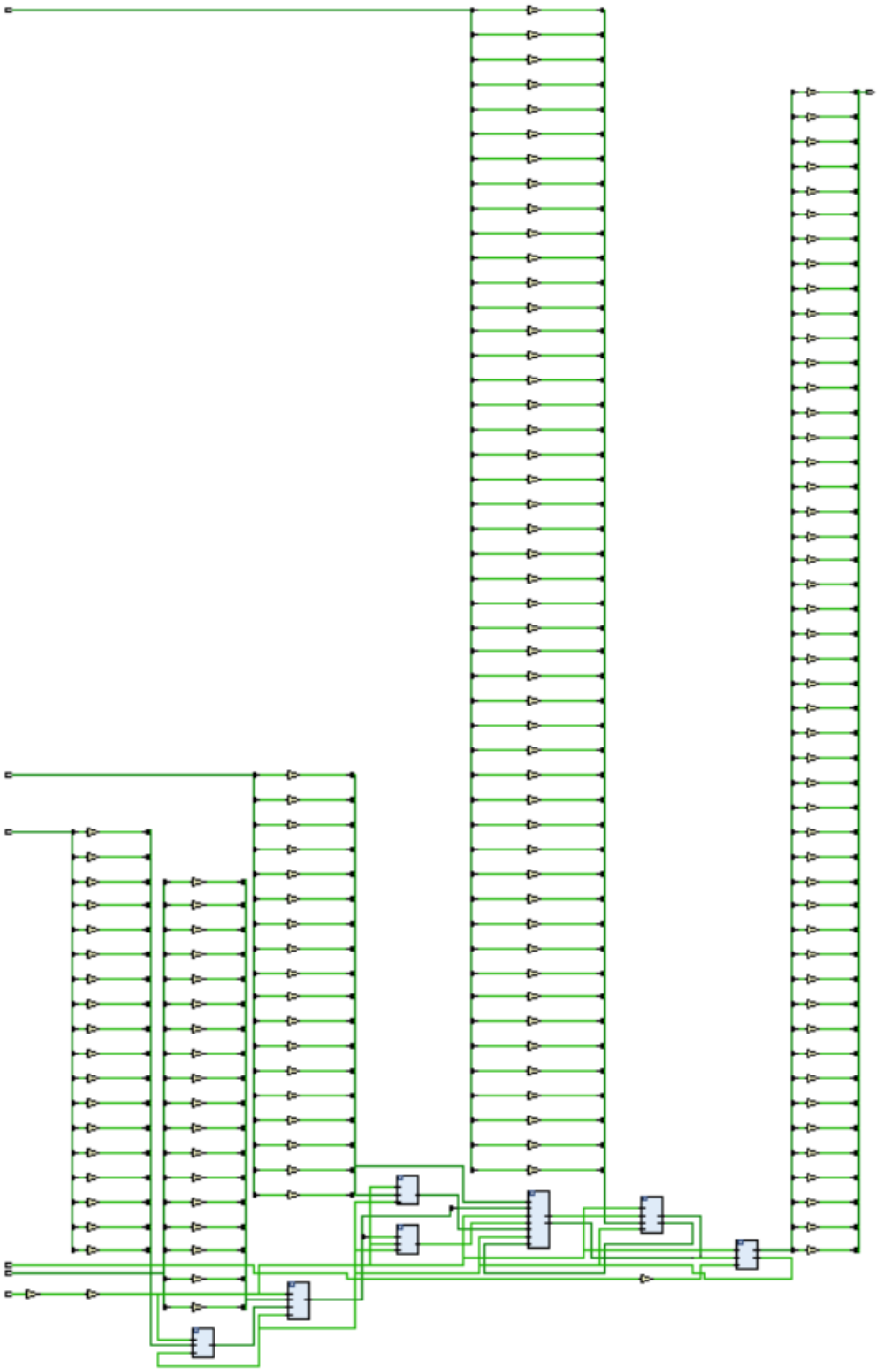


Name	Slice LUTs (134600)	Slice Registers (269200)	DSPs (740)	Bonded IOB (500)	BUFGCTRL (32)
Simple_DSP	55	151	1	152	1
d0 (D_Flipflop)	18	18	0	0	0
d1 (D_Flipflop_0)	0	18	0	0	0
d2 (D_Flipflop_1)	0	18	0	0	0
d3 (D_Flipflop__para...	0	48	0	0	0
d4 (D_Flipflop_2)	0	1	0	0	0
d6 (D_Flipflop__para...	36	0	1	0	0
d7 (D_Flipflop__para...	1	48	0	0	0

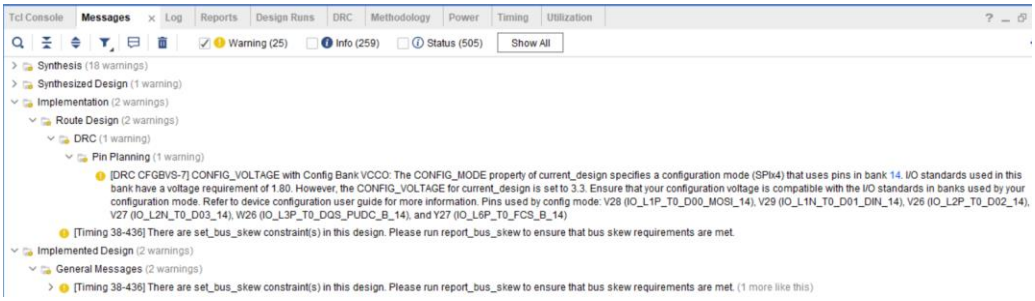
5-synthesis(Timing_report)



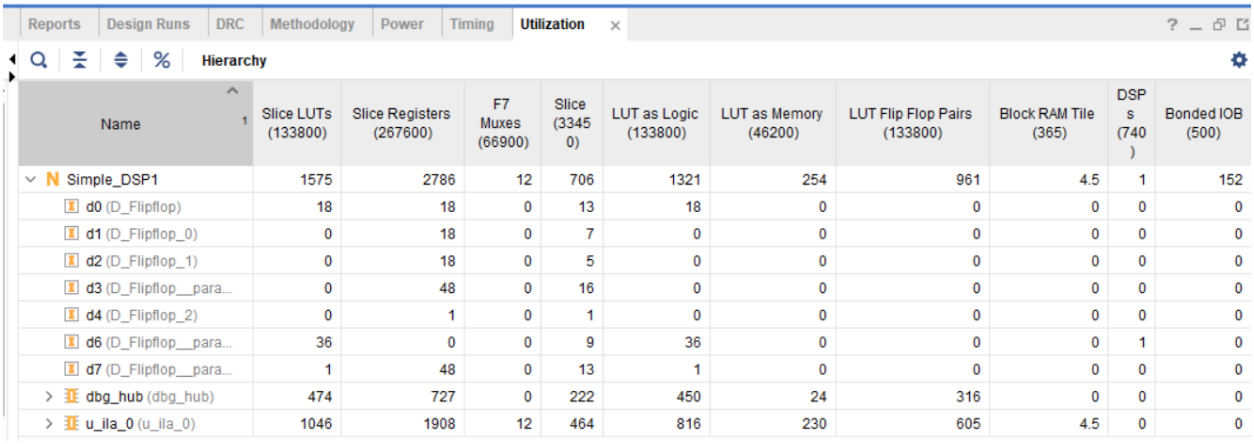
6-synthesis(Schematic)



7-implmentation(messages_tab)



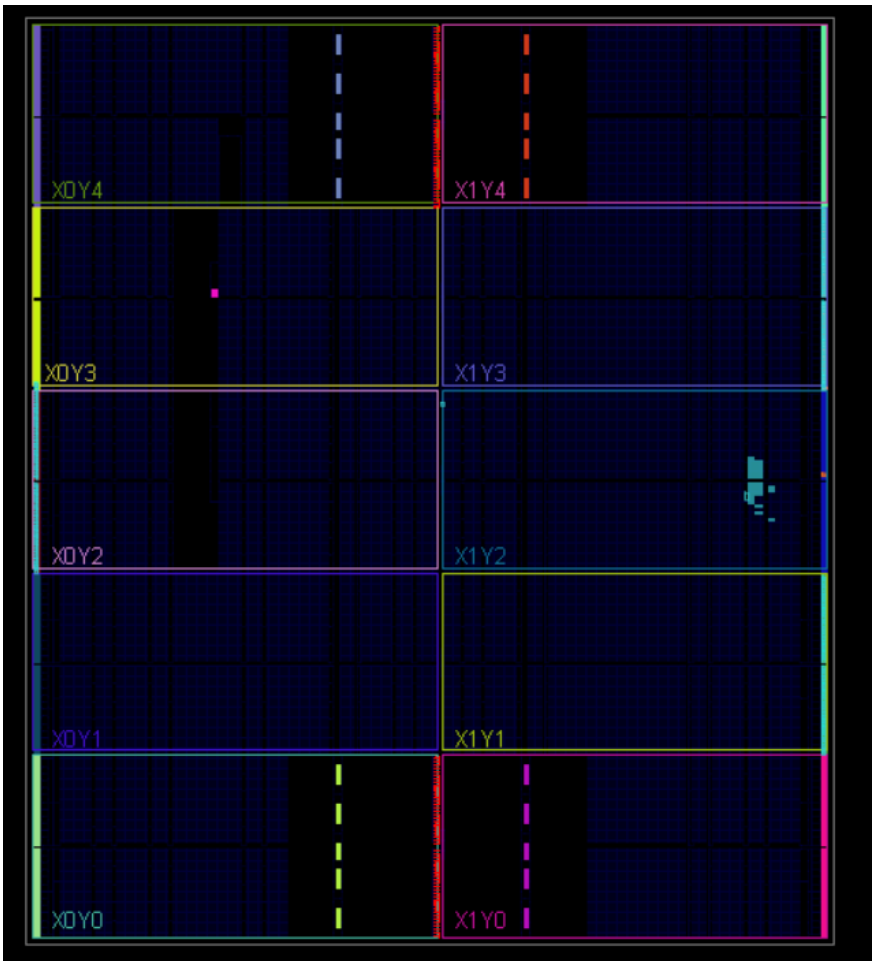
8-Implmentation(Utilization_report)



Name	Slice LUTs (133800)	Slice Registers (267600)	F7 Muxes (66900)	Slice (33450)	LUT as Logic (133800)	LUT as Memory (46200)	LUT Flip Flop Pairs (133800)	Block RAM Tile (365)	DSPs (740)	Bonded IOB (500)
Simple_DSP1	1575	2786	12	706	1321	254	961	4.5	1	152
d0 (D_Flipflop)	18	18	0	13	18	0	0	0	0	0
d1 (D_Flipflop_0)	0	18	0	7	0	0	0	0	0	0
d2 (D_Flipflop_1)	0	18	0	5	0	0	0	0	0	0
d3 (D_Flipflop__para...	0	48	0	16	0	0	0	0	0	0
d4 (D_Flipflop_2)	0	1	0	1	0	0	0	0	0	0
d6 (D_Flipflop__para...	36	0	0	9	36	0	0	0	1	0
d7 (D_Flipflop__para...	1	48	0	13	1	0	0	0	0	0
dbg_hub (dbg_hub)	474	727	0	222	450	24	316	0	0	0
u_ila_0 (u_ila_0)	1046	1908	12	464	816	230	605	4.5	0	0

9-Implmentation(Timing_report)

10-Implmentation(device)



snippets of wave: